**Universidad Central de Venezuela**

**Facultad de Ciencias**

**Escuela de Computación**

**Ingeniería de Software**

**Semestre 1-2017**

**Proyecto**

**Entrega 4:**

**Disciplina de Prueba**

**Equipo 1**

Integrantes:

Alejandra Vento

Laura Zambrano

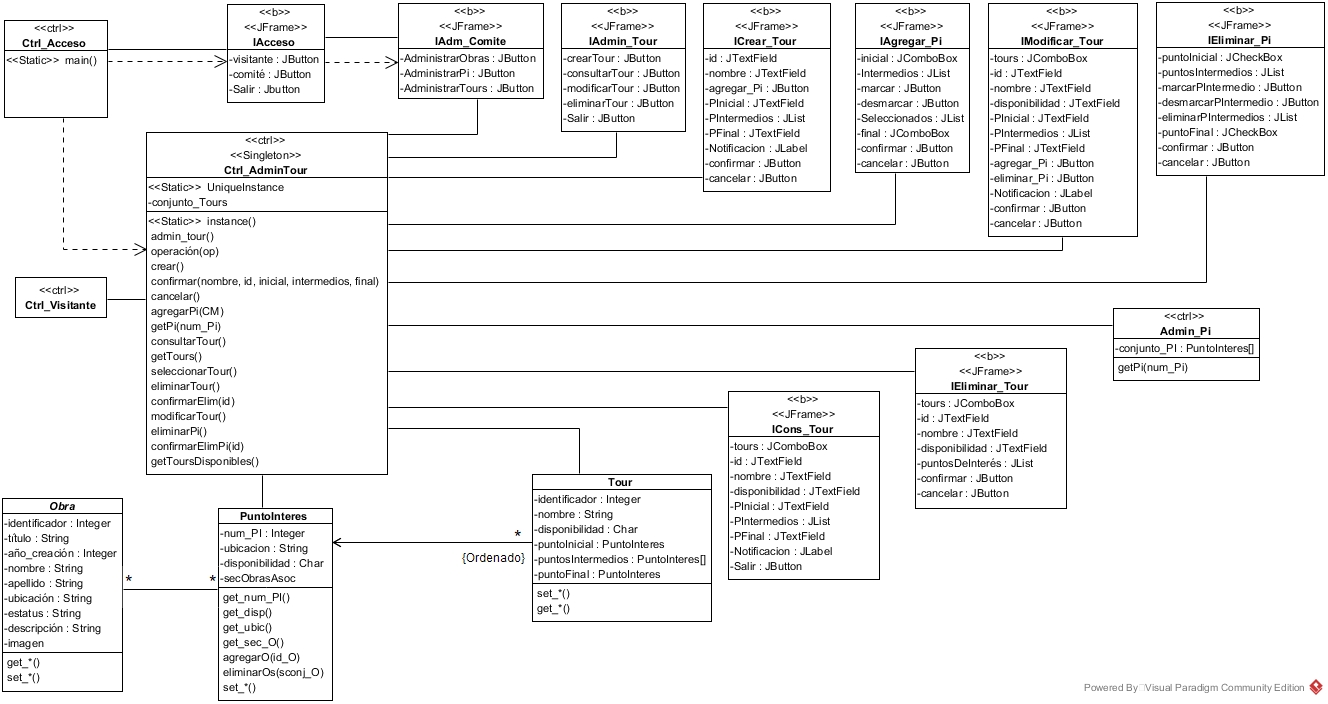
Patrizia Guarente

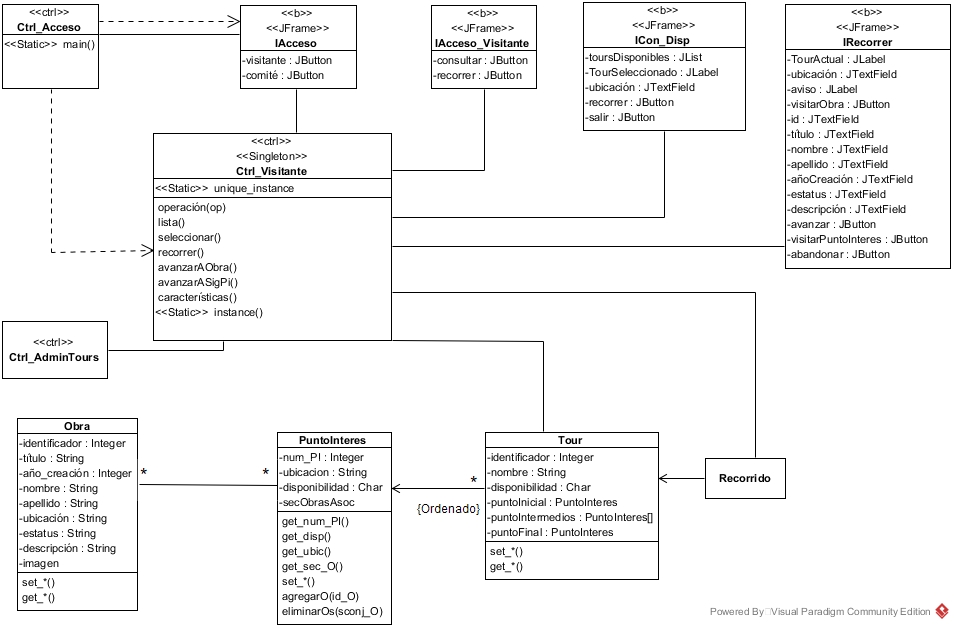
Jesús Martínez

Jesús Pacheco

# Modelo de Prueba

Diagrama de clases de referenica:

**

**

## Especificación de los casos de prueba

**Id Prueba**: 1

**Tipo de Prueba**: Unitaria

**Descripción**: Se crea un tour con sus datos como parámetros recibidos por la función, siempre y cuando el ID dado no coincida con el ID de un tour ya creado

**Clase**: Ctrl\_AdminTour

**Método**: Confirmar(String nombre, int id, int inicial, int intermedios[ ], int final) **Tipo de retorno**: void

**Pre-condición**: -.

**Post-condición**: CARD(Tours) > 0

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2), (t3), (t4), (t5)};

t1= (“Tour1”, 1, 1, {3,29,7}, 5),

t2= (“Tour2”, 1, 15, 29, 1),

t3= (“Tour2”, 2, null, {5,7,15}, 29),

t4= (“Tour3”, 3, 1, null, 5),

t5= (“Tour4”, 4, 5, {15,29}, null)

t6= (“Tour5”, -2, 5, {15,29}, 3)

donde t1 es un caso de prueba para ID de tour válido con Disponibilidad(Tour)=S, t2 es un caso de prueba para ID de tour repetido, t3, t4 y t5 son casos de prueba de tour con Disponibilidad(Tour)=N y t6 es un caso de prueba para ID de tour inválido. Además, para t1 CARD(Tour)=0

**Valor esperado**:

Para T = (t1):

(confirmar (“Tour1”,1, 1, {3,29,7}, 5) :: CARD(Tours) = 1, Disponibilidad(Tours) = S.

Se imprime por consola CARD(Tours), Disponibilidad(Tour) y el mensaje: “Se ha creado el Tour exitosamente”)

Para T = (t2)

(confirmar (“Tour2”, 1, 15, 29, 1) :: CARD(Tours) = 1.

Se imprime por consola CARD(Tours) y el mensaje: “Identificador ya existe”)

Para T = (t3):

(confirmar (“Tour2”, 2, null, {5,7,15}, 29) :: CARD(Tours) = 2, Disponibilidad(Tour) = N.

Se imprime por consola CARD(Tours), Disponibilidad(Tour) y el mensaje: “Se ha creado el Tour exitosamente”)

Para T = (t4):

(confirmar (“Tour3”, 3, 1, null, 5) :: CARD(Tours) = 3, Disponibilidad(Tour) = N.

Se imprime por consola CARD(Tours), Disponibilidad(Tour) y el mensaje: “Se ha creado el Tour exitosamente”)

Para T = (t5):

(confirmar (“Tour4”, 4, 5, {15,29}, null) :: CARD(Tours) = 4, Disponibilidad(Tour) = N.

Se imprime por consola CARD(Tours), Disponibilidad(Tour) y el mensaje: “Se ha creado el Tour exitosamente”)

Para T = (t6):

(confirmar (“Tour5”, -2, 5, {15,29}, 3) :: CARD(Tours) = 4.

Se imprime por consola CARD(Tours) y el mensaje: “ID inválido”)

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 2

**Tipo de Prueba**: Unitaria

**Descripción**: Se elimina el Tour seleccionado del conjunto Tours siempre y cuando CARD(Tours) > 0

**Clase**: Ctrl\_AdminTour

**Método**: confirmarElim (int id) **Tipo de retorno**: void

**Pre-condición**: CARD(Tours) > 0

**Post-condición**: CARD(Tours)=CARD(Tours)’-1, CARD(Tours)≥0

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2)};

t1= (1)

t2= (1)

donde t1 es un caso de prueba de eliminar un tour válido y t2 es un caso de prueba de eliminar un tour inválido. Además, para t1 CARD(Tours) > 0

**Valor esperado**:

Para T = (t1):

(confirmarElim (1) :: CARD(Tours) = CARD(Tours)’-1

Se imprime por consola CARD(Tours), el conjunto de Tours y el mensaje: “Se ha eliminado el Tour exitosamente”)

Para T = (t2):

(confirmarElim (1) :: CARD(Tours) = CARD(Tours)’

Se imprime por consola CARD(Tours) y el mensaje: “El Tour seleccionado no existe”). Si se utiliza la interfaz este caso de prueba no va a ocurrir

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 3

**Tipo de Prueba**: Unitaria

**Descripción**: Se informan los tours contenidos en el conjunto Tours, siempre y cuando CARD(Tours) > 0

**Clase**: Ctrl\_AdminTour

**Método**: seleccionarTour(id) **Tipo de retorno**: ArrayList

**Pre-condición**: CARD(Tours)≥0

**Post-condición**: -.

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2)};

t1= (1)

t2= (-12)

donde t1 es un caso de prueba de consultar un tour válido y t2 es un caso de prueba de consultar un tour inválido. Además, para t1 CARD(Tours) > 0

**Valor esperado**:

Para T = (t1):

(seleccionarTour (1) :: Si se imprimen las siguientes variables: ID, Nombre, inicial, intermedios[ ], final; se pondrían aquí?

Se imprime por consola ID, Nombre y los puntos de interés asociados)

Para T = (t2):

(seleccionarTour (123) :: ( solo se muestra el mensaje, que podemos poner aquí

Se imprime por consola el mensaje: “El Tour seleccionado no existe”). Si se utiliza la interfaz este caso de prueba no va a ocurrir

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 4

**Tipo de Prueba**: Unitaria

**Descripción**: Se asocian los puntos de interés seleccionados al tour previamente escogido

**Clase**: Ctrl\_AdminTour

**Método**: agregarPi (bool CM, int id, int inicial, int intermedios [ ], int final) **Tipo de retorno**: bool

**Pre-condición**: -.

**Post-condición**: CARD(Tours)>0

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2), (t3), (t4), (t5), (t6)};

t1= (1,1,-12, null, 5)

t2= (1,3, 1, {3, -4, 7}, 29)

t3= (1,2, 15, {3, 7}, -9)

t4= (1,1, 501, 3, 7)

t5= (1,4, 1, {3,5}, 7)

t6= (1,4, 1, null, 7)

donde t1, t2 y t3 son casos de prueba de un ID de punto de interés inválido, t4 es un caso de prueba de un ID de un punto de interés inexistente, t5 es un caso de prueba de un punto de interés válido, t6 es un caso de prueba de un conjunto de puntos de interés vacío. Además, para t5 CARD(conjunto\_PI) > 0 y para t6 CARD(conjunto\_PI) = 0

**Valor esperado**:

Para T = (t1):

(agregarPi (1,1,-12, null, 5) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t2):

(agregarPi (1,3, 1, {3, -4, 7}, 29) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t3):

(agregarPi (1,2, 15, {3, 7}, -9) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t4):

(agregarPi (1,1, 501, 3, 7) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “El punto de interés no existe”)

Para T = (t5):

(agregarPi (1,4, 1, {3,5}, 7) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “Los puntos de interés fueron agregados exitosamente”)

Para T = (t6):

(agregarPi (1,4, 1, null, 7) :: CARD(conjunto\_PI) = 0

Se imprime por consola el mensaje: “No hay puntos de interés”)

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 5

**Tipo de Prueba**: Unitaria

**Descripción**: Se eliminan los puntos de interés seleccionados del tour previamente escogido

**Clase**: Ctrl\_AdminTour

**Método**: confirmarElimPi(int id, int inicial, int intermedios [ ], int final) **Tipo de retorno**: void

**Pre-condición**: CARD(Tours) > 0

**Post-condición**: CARD(Tours) > 0

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2), (t3), (t4), (t5), (t6)};

t1= (1,-12, null, 5)

t2= (3, 1, {3, -4, 7}, 29)

t3= (2, 15, {3, 7}, -9)

t4= (1, 501, 3, 7)

t5= (4, 1, {3,5}, 7)

t6= (5, null, null, null)

donde t1, t2 y t3 son casos de prueba de un ID de punto de interés inválido, t4 es un caso de prueba de un ID de un punto de interés inexistente, t5 es un caso de prueba de un punto de interés válido, t6 es un caso de prueba donde no se seleccionen puntos de interés previos. Además, para t5 y t6 CARD(conjunto\_PI) > 0

**Valor esperado**:

Para T = (t1):

(confirmarElimPi (1,-12, null, 5) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t2):

(confirmarElimPi (3, 1, {3, -4, 7}, 29) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t3):

(confirmarElimPi (2, 15, {3, 7}, -9) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t4):

(confirmarElimPi (1, 501, 3, 7) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “El punto de interés no existe”)

Para T = (t5):

(confirmarElimPi (4, 1, {3,5}, 7) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “Los puntos de interés fueron eliminados exitosamente”)

Para T = (t6):

(confirmarElimPi (5, null, null, null) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “No han sido seleccionados los puntos de interés”)

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 6

**Tipo de Prueba**: Unitaria

**Descripción**: Se muestran y seleccionan los puntos de interés a eliminar de un tour

**Clase**: Ctrl\_AdminTour

**Método**: eliminarPi (int id) **Tipo de retorno**: void

**Pre-condición**: CARD(Tours) > 0

**Post-condición**: CARD(Tours) > 0

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2)};

t1= (4)

t2= (501)

t3= (-2)

t4= (3)

donde t1 es un caso de prueba para eliminar puntos de interés válidos, t2 es un caso de prueba para ID inexistente, t3 es un caso de prueba para ID inválido, t4 es un caso de prueba donde no existen puntos de interés para eliminar. Además, para t1 CARD(Tours) > 0

**Valor esperado**:

Para T = (t1):

(eliminarPi (4) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola los puntos de interés asociados al tour)

Para T = (t2):

(eliminarPi (501) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “ID del tour inexistente”)

Para T = (t3):

(eliminarPi (-2) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “ID del tour inválido”)

Para T = (t4):

(eliminarPi (3) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “El tour no posee puntos de interés”)

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 7

**Tipo de Prueba**: Unitaria

**Descripción**: Se obtienen todos los tours disponibles para recorrer

**Clase**: Ctrl\_Visitante

**Método**: lista() **Tipo de retorno**: ArrayList

**Pre-condición**: -.

**Post-condición**: -.

**Casos de prueba**: Sea T ⊆D y T ={(t1),(t2)};

t1= ()

t2= ()

donde t1 es un caso de prueba que obtiene los tours disponibles para recorrer y t2 es un caso de prueba que obtiene todos los tours disponibles sin importar su disponibilidad.

**Valor esperado**:

Para T = (t1):

(lista()) :: (solo se muestra el mensaje, que podemos poner aquí?)

FALTA

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 8

**Tipo de Prueba**: Unitaria

**Descripción**: Se obtiene el siguiente punto de interés del tour para recorrer

**Clase**: Ctrl\_Visitante

**Método**: avanzarASigPi(Tour t, int ID) **Tipo de retorno**: PuntoInterés

**Pre-condición**: -.

**Post-condición**: -.

**Casos de prueba**: Sea T ⊆D y T ={(t1),(t2),(t3)};

t1= (tour1,1)

t2= (tour2,3)

t3= (tour3,5)

donde t1 es un caso de prueba que obtiene el siguiente punto de interés el cual tiene disponibilidad N, t2 es un caso de prueba que obtiene el siguiente punto de interés el cual tiene disponibilidad S y t3 es un caso de prueba que obtiene el siguiente punto de interés el cual no existe.

**Valor esperado**:

Para T = (t1):

(lista(tour1,1)) :: (solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “el punto de interés no está disponible”

Para T = (t2):

(lista (tour2,3) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se muestra el punto de interés para recorrer.

Para T = (t3):

(lista (tour3,5) :: ( solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “el punto de interés no existe”

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 9

**Tipo de Prueba**: Unitaria

**Descripción**: Se obtienen todos los tours disponibles para recorrer

**Clase**: Ctrl\_Visitante

**Método**: avanzarAObra(String nombre, String ID) **Tipo de retorno**: Obra

**Pre-condición**: -.

**Post-condición**: -.

**Casos de prueba**: Sea T ⊆D y T ={(t1),(t2)};

t1= (“Pastor de nubes”,”OA-ESC-13”)

t2= (“La Mona Corrugosa”, OE-ELC-15”)

donde t1 es un caso de prueba obra valido y t2 es un caso de prueba obra invalido

**Valor esperado**:

Para T = (t1):

(avanzarAObra(“Pastor de nubes”,”OA-ESC-13”) :: (solo se muestra el mensaje, que podemos poner aquí?)

Se muestra por pantalla la obra seleccionada)

Para T = (t2):

(avanzarAObra(“La Mona Corrugosa”, OE-ELC-15”) :: (solo se muestra el mensaje, que podemos poner aquí?)

Se imprime por consola el mensaje: “La obra no existe”)

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

## Diagrama de paquete

*Nota: explicar aquí la decisión que se decidió tomar para organizar las clases en paquete, por ejemplo si se usa el criterio MVC, etc.*

*Nota: Incluir diagrama hecho en visualParadign*